11 优化

11.1 Introduction

创作实时项目的遇到挑战是难以预料的。除了所有的硬件与软件，还要有极为精准的实时数据。 当项目运行的 FPS 为 30 时，每次的处理、显示与创建必须在 33 毫秒时间内完成。这甚至不到十分之一秒。当 FPS 越高时，这个时间越小。运行在 FPS 为 60 的项目，每帧的渲染从开始至结束只有 16ms。

意识到这是多么短的时间，因而珍惜每一毫秒的每一部分是多么重要。要清楚的知道元件的烘焙为何要花费一毫秒，慎重处理，尽可能地节省时间，知道每一毫秒都是有用的。而要做到这些需要项目分析与优化的基础技能。

TouchDesigner 使用到大量的 CPU 与 GPU 资源，能够找出是哪里需要这么多的资源是很重要的技能。当面对大型网络时，知道系统在哪里停滞，以及如何优化元件避免掉入这些陷阱，将决定项目能否成功交付。

11.2 找到瓶颈

将计算机看作一条流水线，里面的 CPU、GPU、内存与硬盘共同生产最终的产品。它们单独执行非常具体的任务，因此有时单独工作，但通常它们之间互相合作。在一条流水线上，系统运行速度只能和最慢的地方一样快。因为这种依赖性，即便其他的环节都是完全正常的，这个环节也会拖慢整个项目。这种在流水线中的薄弱环节，被称为瓶颈。

举一个瓶颈的案例，在项目中要渲染一些基础的三维几何体，为它们的表面贴上视频素材。假设这个项目包含 4 个 Box SOP，在它们的每个面贴上 1920*1080 HAP Q 格式的影片素材。电脑配置 32G 内存，8 核双芯 CPU，顶级的 NVIDIA Quadro 显卡和一块 5400-RPM 硬盘。

启动后，在这样的电脑系统上项目是运行不起来的。不管有多少内存、多少 CPU，以及多么昂贵的显卡，这个项目不能从 5400-RPM 硬盘中一次性读取多个 HAP Q 文件。因为硬盘转速不够快，不能同时读取多个 HAP Q 文件，电脑将停滞。HAP Q 文件对硬盘的要求比较高，无论电脑的其他部分多么强大，这个项目也不会运行。如硬盘不能处理像素一样，GPU 不能从硬盘读取文件。在这种情况下，硬盘就成为这个项目的瓶颈。

通常，瓶颈会出现在这三个地方：GPU、CPU 和硬盘。

GPU 本身就是一个流水线，像素着色阶段很可能成为瓶颈。每当操作像素，使用几乎所有 TOP，系统需求更多的 GPU 着色器。越高的分辨率，越高的 GPU 需求。TOP 的分辨率与 GPU 运算量呈 1:1 的正比关系。如果 TOP 分辨率降低一半，它的 GPU 运算量也会按比例降低。是否有像素着色器瓶颈的一个快速检查方法是降低所有生成 TOP（如 Render TOP 与 Constant TOP）的分辨率。如果在速度与性能上有明显的增加，显然存在像素着色器瓶颈。

当显卡过载，看似随意的 TOP 将开始耗费比普通渲染更长的时间。在 Performance Monitor 中，TouchDesginer UI 元素的渲染时间可明显观察到。如果所有的 UI 元素的渲染时间突然超过 1 毫秒，就需要优化网络减轻 GPU 在这一块的负担。

从经验来看，CPU 是瓶颈出现的第二个地方。所有的元件都需要 CPU 运行，因此 CPU 很快会过载。可以在 Performance Monitor 可以看到元件消耗大量的时间渲染，因而 CPU 的瓶颈往往比较容易跟踪。用 Hog CHOP 可以衡量有多少 CPU 的预制时间（headroom）。这个 CHOP 的作用与其名称一样，占用 CPU 处理。其“Delay”参数设置每一帧渲染总时间。在创建 Hog CHOP 后，项目开始丢帧，就说明 CPU 是我们的瓶颈。

所有的影片文件使用 CPU 从它们的压缩状态解码数据。某些编码器对 CPU 的使用比其他多，读取多个影片文件时使用的 CPU 资源比想像中的多。

过载的 CPU 反应类似于过载的 GPU，在 Performance Monitor 会出现不一致的结果。元件呈现不同的烘焙时间。这是因为它们开始操作后，在它们完成之前，CPU 又被叫去执行其他过程。 元件花费时间等待 CPU 回来执行它的过程因而增加它的烘焙时间。与 GPU 过载不同，CPU 过载不仅仅影响 TOP。

硬盘驱动器是瓶颈出现的第三个地方。 特定操作会对硬盘的要求比较高。当为部署项目配备系统时容易忽视高质量的固态硬盘（SSD）。取决于使用的编码器，在硬盘上读写影片的操作会快速耗尽硬盘性能。这样的瓶颈提示会出现在 Performance Monitor 中 Movie In TOP 行下，如：

```
Waiting for frame for 90 - E:/test.mov from harddrive
```

数字是影片的帧，路径是影片的文件。

11.3 Using Performance Monitor

Performance Monitor 是分析帧烘焙时间的工具。当要进行优化或调试项目性能时很有用。

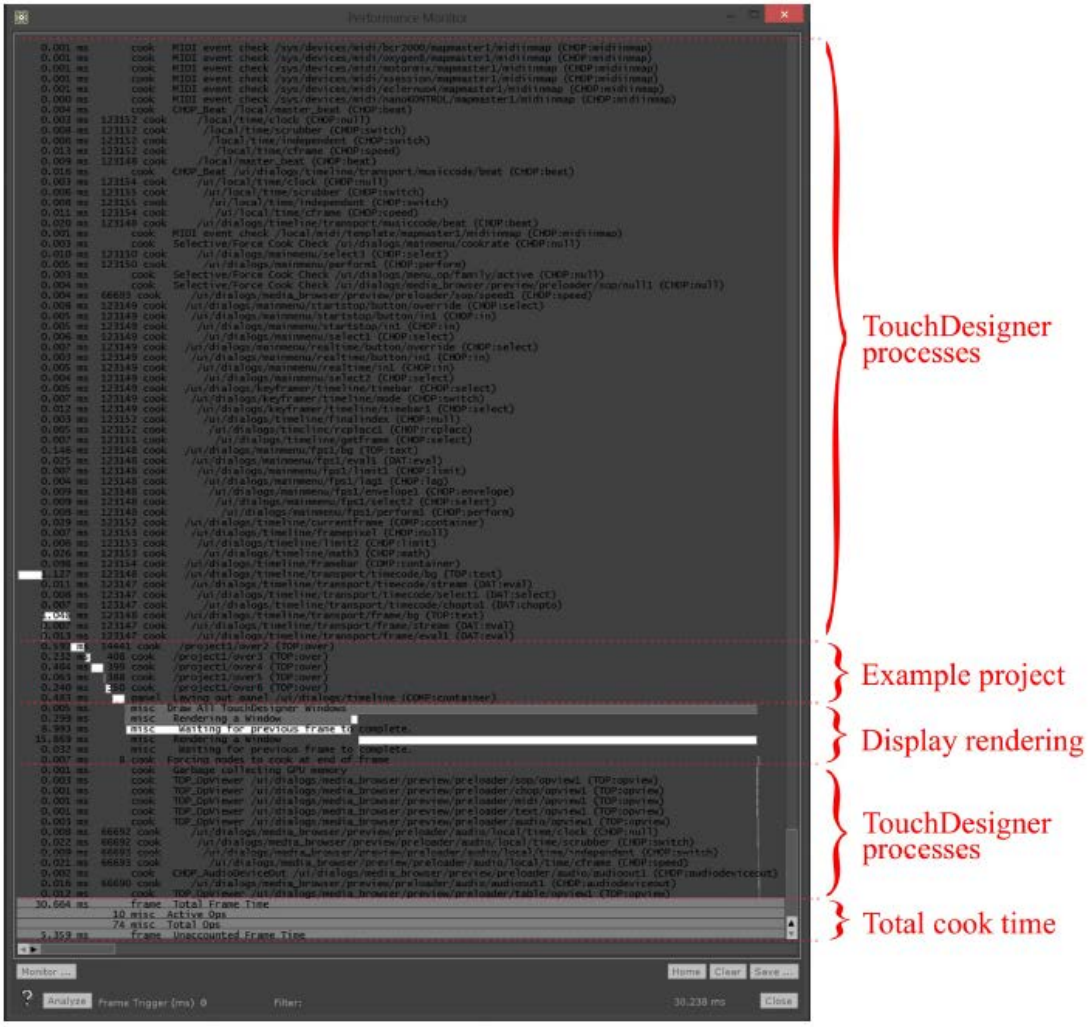
有三种方法访问 Performance Monitor。
1. 键盘上按“F2”
2. 快捷键 “Alt + Y”
3. 在屏幕顶部菜单“Dialogs”下拉列表中选择“Performance Monitor”

它有几个按钮，功能一目了然。

1. 执行分析当前帧
2. 清除当前结果
3. 保存当前结果至文件文件
4. 改变监视选项
5. 毫秒阈值，如果烘焙时间到达，触发 Performance Monitor 进行分析
6. 更加精确地筛选结果，如只查看 CHOP 或 TOP 等。

需要注意的是，烘焙时间基于 CPU，并不意味着 GPU 瓶颈会在 Performance Monitor 中被忽视，但是在 Monitor 中我们要意识到这是 CPU 的数据。

这是一个来自 Performance Montior 分析结果的例子：



上面的分析来自一个简单的项目，它包含几个 Movie In TOP 与几个 Over TOP。这些元件负责 TouchDesigner 的功能和用户界面元素。

在上图标记的部分查看渲染时间与路径，TouchDesigner Processes 中的元件负责 TouchDesigner 的主要功能（用户界面、会话窗口等等），而属于工程案例（Example Project）的元件更加重要。每个元件烘焙时间用白条代表。白条的尺寸代表它占用总烘焙时间的比例。当项目增长变得更加复杂时，“Example project”标记块会相应增长并包含项目中所有的元件。通过它来分析项目，找到占用太多系统资源的问题元件。

在这个案例中，可以跟踪 Over TOP 的路径与烘焙时间。

它们都在“project1”容器内，其渲染时间范围在 0.06 毫秒到 0.6 毫秒之间。

强烈提醒：注意垂直同步的影响，以及渲染时间在 Performance Monitor 是如何被计算的。显卡会尝试锁定项目的 FPS 与显示刷新率。在 30FPS，即使是一个空的项目，每帧的烘焙时间也是 33 毫秒。在 60FPS，会有同样的效果，Performance Monitor 将会显示烘焙时间为 16 毫秒。这并不是说每帧的实际渲染时间刚好是 33 毫秒或 16 毫秒，而是垂直同步尝试同步 TouchDesigner 的 FPS 与显示刷新率。

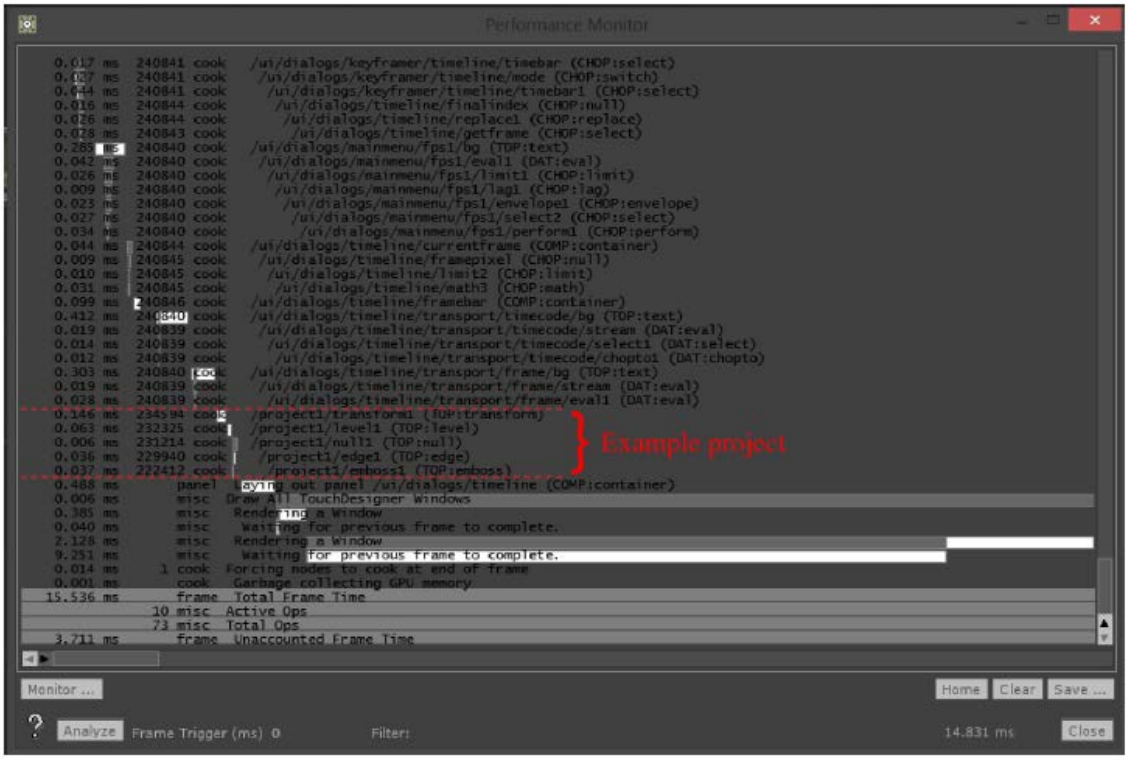
11.4 Operator Cooking

通过降低元件每帧的烘焙总时间以得到更好的性能。初学者在创建网络时可能从来不考虑到这一点。诚然，每个人都有赶工的时候，在创建网络时无法考虑太多。然而，如果一个项目要求不能丢帧 ，那么不考虑烘焙是非常冒险的。

主要的目标是在信息流程中尽可能早的处理静态操作以防止在每一帧都被渲染。

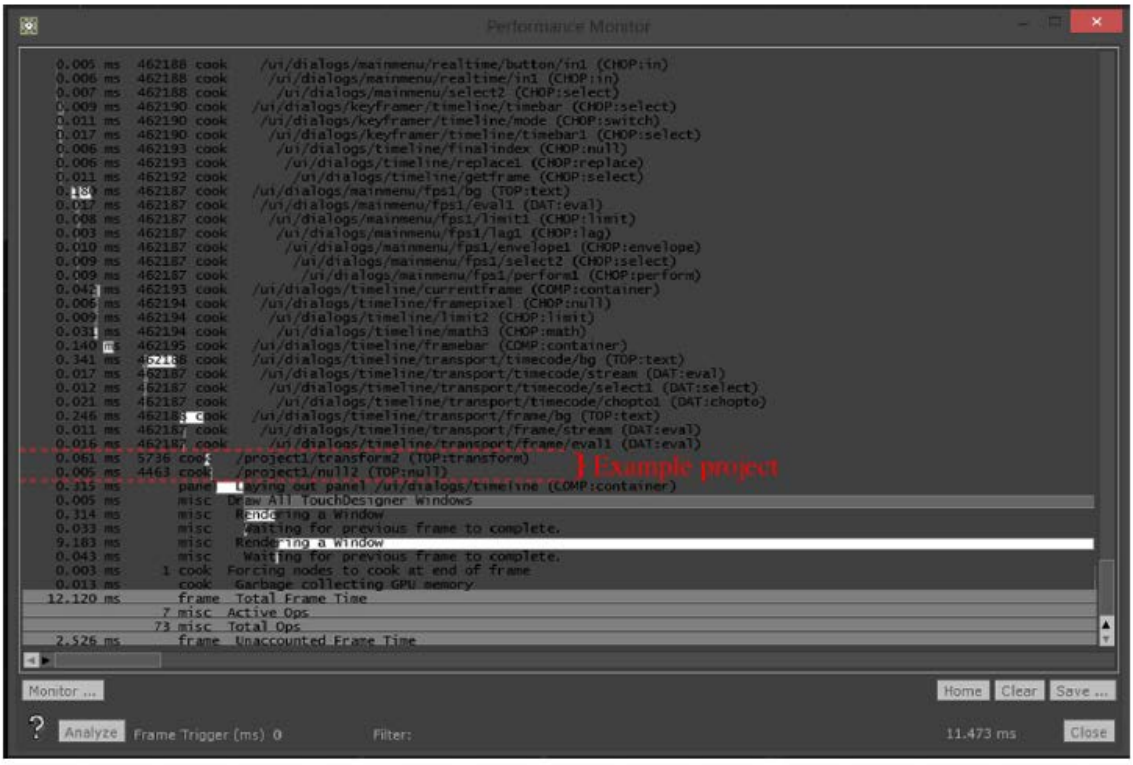
打开“Cooking_1.toe”。在这个案例，旋转一个简单的图形，然后再给它其他操作使得看起来更有趣。有一个功能可以帮助优化网络：连接元件的带动画的线。动画的线意味着线的两端每帧都会被烘焙。从网络的最左边开始，连接 Movie In TOP 与 Transform TOP 的线是没有动画的。这是因为图像被载入并保持静态。元件只在它们需要被执行或改变时才进行烘焙。一个静止的图片是静态的，不做改变，也就不需要每帧都进行烘焙。

与之相反，网络中的其他线都是带动画的，意味着 Movie In TOP 之后的所有元件在每一帧都需要烘焙。对于这个项目，这不是一个可怕的问题，因为这里没有发生非常复杂的事情。从一开始就创建高效的网络可以节省优化网络提升性能的时间。让我们在 Performance Monitor 中查看这个项目。



忽略为实现 TouchDesigner 功能必要的所有元件，有一小块展示的是这个案例。对图像执行的操作，总共花费了大约 0.25 毫秒。如前所述，静态操作只烘焙一次，要注意的是许多操作在 Transform TOP 之后都是静态的。让我们重新排列这些，再查看性能上的提升。

打开“Cooking_2.toe”。这个项目与之前的项目完全相同，除了元件被重新排列，在查看信息流程之前，让我们先来查看 Performance Monitor。



乍一看，似乎这个项目缩小了。有几个元件已经消失。因为这些元件并不是每帧都要烘焙。在上一个案例中，Transform TOP 在每帧都执行变换，使得它之后的所有 TOP 每帧都需要重新计算。而在这个案例中，所有的静态操作都放在信息流程的开始，留下 Transform TOP 执行它的变换，其他元件都不需要重新计算或烘焙。

更进一步查看 Performance Monitor 结果，只有 Transform TOP 元件被烘焙，而且耗费是 0.061 毫秒。

比较之前的案例，一系列的元件烘焙耗费 0.25 毫秒。简单的改变却有令人难以置信的改善。

值得注意的，两组元件的输出结果可能会不完全相同，但是如此难以区分，以至于许多客户和艺术家都不会介意，尤其在知道有这么大的性能提升后会允许这样做。

11.5 分辨率

在谈到平面纹理时，分辨率是非常重要的。因为每一个像素点的消耗都会带来资源的浪费。

用一个案例以一个非常很简单的方式说明这一事实。打开“Resolution_1.toe”。有这样的一个简单的设置：将一只蝴蝶连在一个大画布上，然后使用一些 LFO（低频振荡器），在合成至森林背景之前调整它的透明度与模糊。在任意一个 TOP 上点击中会显示这个案例需要超过 100M 的 GPU RAM。在一个有 4G RAM 的 GPU 系统中这不是一个很大的空间，但是它会速度地增加。尝试实时合成 40 只蝴蝶 ，4G 空间很容易耗光。

现在对比“Resolution_2.toe”。它得到完全相同的结果，但是只消耗了 60M 的 GPU RAM。这是一个显著的差异。使用第二种方法，在案例中合成 40 只蝴蝶，只占用大约 2.4G 的 GPU 内存。由此可见产生的空间差异仅仅是因为我们做出了一些小的改变。

蝴蝶的源资产只有 512*512 的分辨率，在第一个案例中，立刻合成到一块 1920*1080 的画布上，然后调整。蝴蝶被调整后，Touchdesigner 需要在每一帧重画 1920*1080 的画布。“空”像素即使没有颜色与 Alpha 数据也同样被重画。在第二个案例，执行完全相同的操作，但只在更低分辨率的源资产上操作。把调整后的资产合成在 1920*1080 的画布。这样 GPU 只需要处理一小部分像素，节省在大画布上的重画，从而节省相当多的 GPU 内存与处理能力。

11.6 GPU 内存碎片

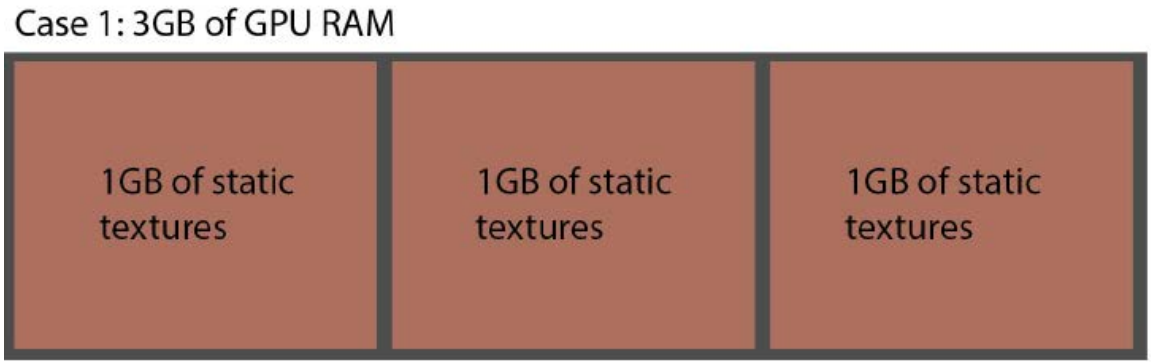
使用 GPU 的元件通常为它们的任务分配资源，并保持占用，直到任务完成或改变。

当运行在有多个不同内容的来源项目运行时，GPU 碎片是一个主要的关注点。举例，一个 1280*720 的 Constant TOP 元件连接到 10 个其他元件上。一旦连接并烘焙，各元件将预留出需要处理特定数量像素的 GPU 内存。一旦内存被分配，元件能够在它们的分配空间内比较高效的运行。

如果源 Constant TOP 的分辨率发生改变，这将触发一系列的连锁反应，其他 10 个元件全部需要为它们的任务重新分配 GPU 资源。如果元件的资源可以被高效分配，许多一样的内存块将被重用。相反地，如果它们不能重用内存块，而必须迁移至内存尾部。如果这样的情况接二连三的发生，GPU 内存将变得破碎，GPU 将试图整理它的内存，而项目将在此不佳的状态下运行。

下面两幅图尝试以最简单的方式描述内存碎片。

要研究的两个案例，具有相似的参数。3GB 的 GPU RAM，载入 3 张 1G 的静态纹理，并且在 RAM 中一直保持占用。



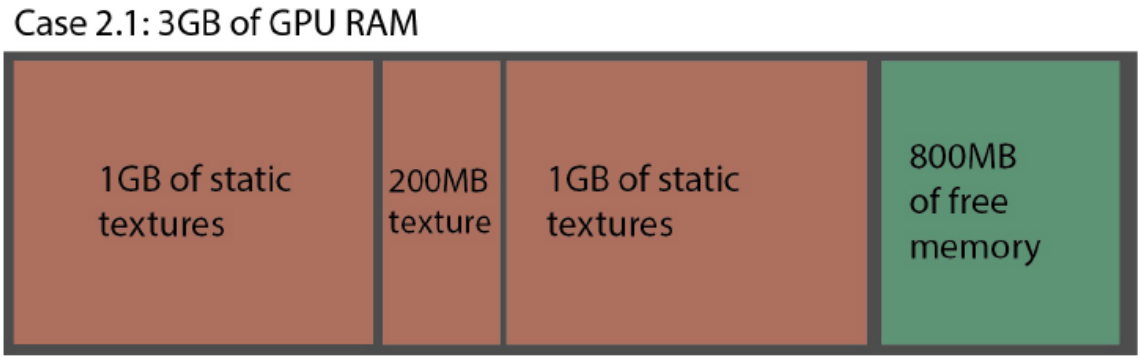
在案例 1 中，3G RAM 的 GPU 可以完全适合三张 1G 的静态纹理。它们是静态纹理因为一旦被分配空间，就不再改变。这相当于从项目一开始就在 Movie In TOP 中载入一张大尺寸图像，然后一直就保持占用。

这是一个理想中的情况，因为有许多进程要使用 GPU 内存资源，意味着它是不断变化的，而且在 3G RAM 的显卡上永远不会有 3G 待分配空间。

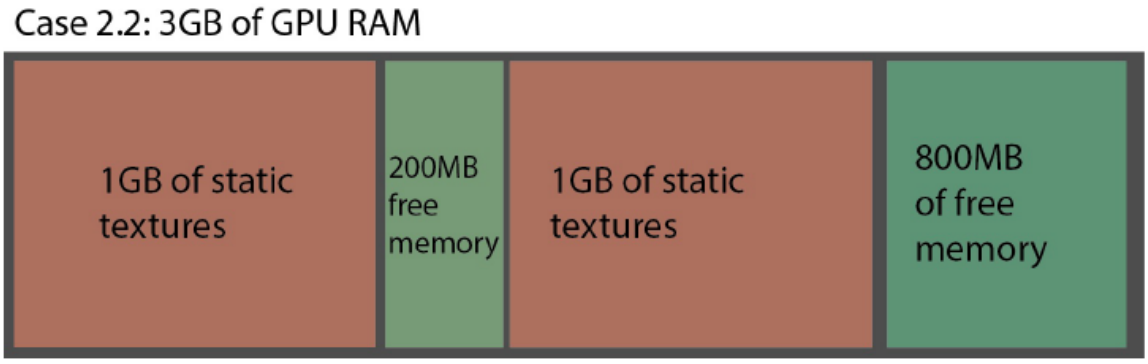
案例 2 描述的情况将产生内存碎片。已经存在三张 1G 的纹理，再加入一张 200M 的纹理。在这个案例中，依如下顺序执行载入与卸载操作。

1. 载入 1G 的纹理
2. 载入 200M 的纹理
3. 载入 1G 的纹理
4. 卸载 200M 的纹理
5. 载入 1G 的纹理

模拟这样的情况： 载入与显示一张纹理，然后使用其他纹理将其替换。



在图 Case 2.1 中，从第一步至第三步，还有 800M 的可用空间。乍一看，这似乎是完美的，因为 200M 的纹理被卸载，有足够载入最后纹理的 1G 空间。不幸的是，这不是显卡的工作方式。



如上所述，在图“Case 2.2”中，第四步完成后，200M 纹理被卸载。这是 GPU 内存碎片的典型例子。总共有 1G 的 GPU 可用空间，但是没有一块完整的 1G 的空间以分配给 1G 的纹理。已经载入的 1G 纹理，它们处理静止状态，不能在 GPU 内存没有完全卸载和重载的过程中发生迁移。因为内存不能移动，200M 的可用空间会一直被困在静态纹理之间，这 200M 的可用空间可以被 200M 或更小的纹理使用，但不能载入第三张 1G 的纹理。

防止内存碎片严重的最好方法是尝试与限制项目中使用不同分辨率的数量。如果换成相同分辨率的资产，通常可以使用它在内存中的原来位置。

11.7 Windows System Processes

Windows 是一个复杂的系统，有许多进程和与系统相关的程序在后台运行。许多进程和程序会对计算机性能和 TouchDesigner 产生负面影响。

Windows 对于那些白天使用电脑的人有巨大的市场。正因为如此，如果一台电脑在夜间供电，有许多应用程序和 Windows 系统操作将按计划运行。对于一天 24 小时运行的演出装置，这可能会有问题，许多不同的应用程序与系统相关任务应当被关闭或重新安排。

像杀毒软件和间谍扫描软件，可以在日常的电脑中使用，通常应当避免在 TouchDesigner 的电脑中使用。它们可能会导致一些问题，第一个是弹出窗口。许多这样的软件在一定的时间间隔会弹出提醒和通知，如果比较不幸，它们会重叠在显示的内容上并覆盖演出装置的输出。这些程序对硬盘同样会产生负面的影响，因为它们经常扫描系统中的病毒与恶意软件，占用硬盘读写周期。这些都是会占用 CPU 资源的，所以需要注意那些一直在运行的程序。